Heartbleed

Sistemas Operativos

03 de noviembre de 2016

Equipo: Quiñones Rivera Josué Emanuel.
Vázquez Álvarez Ángel Eduardo.
Profesor: Gunnar Eyal Wolf Iszaevich

1. Introducción

El término Heartbleed se usa para nombrar un error de seguridad descubierto en la biblioteca criptográfica del popular software OpenSSL. Heartbleed estuvo presente en el software desde su versión 1.0.1 (lanzada el 21 de marzo de 2012), posteriormente se hizo público el 2 de abril de 2014 y fue finalmente corregido en la versión 1.0.1g (lanzada el 7 de abril de 2014)[1].

El nombre Heartbleed explica por sí solo la naturaleza del error. "Heart" hace referencia al protocolo Heartbeat implementado en OpenSSL y "Bleed" se refiere a la fuga de datos ocasionada por una mala implementación de ese protocolo[2]. En esencia, el error permitía que cualquier persona en Internet pudiese leer el contenido en memoria de sistemas "protegidos" por versiones vulnerables de OpenSSL, es decir, un atacante podía tener acceso a llaves criptográficas, nombres de usuario, contraseñas y cualquier otro tipo de datos privados. Además, para empeorar aún más la situación, el error no dejaba ningún rastro en el dispositivo afectado.

2. Conceptos básicos

El rol que ocupan las computadoras y el Internet en nuestra vida diaria ha hecho que la protección de datos se vuelva una cuestión importante a tratar. El cifrado es simplemente la traducción de datos a un código secreto, de manera que el contenido sólo pueda ser leído usando una llave secreta o contraseña que permita traducir ese código secreto a su estado original. Por esta razón, el cifrado puede ser considerado como una forma efectiva de garantizar la seguridad de información confidencial.

El protocolo más utilizado para la transmisión de datos a través de Internet es HTTP (*Hypertext Transfer Protocol*). Sin embargo, si se requiere cifrar esta información a través de Internet (por ejemplo, al realizar una transacción bancaria), el protocolo más famoso y ampliamente utilizado es HTTPS. HTTPS no es otra cosa que HTTP sobre una capa de SSL/TLS[2]. Lo anterior es relevante debido a que Heartbleed surgió a partir de un error de programación en las bibliotecas de OpenSSL que proveen servicios criptográficos como SSL/TLS a otras aplicaciones y servicios[3].

2.1. ¿Qué es OpenSSL?

OpenSSL es un proyecto colaborativo open source que tiene por objetivo el desarrollo de una biblioteca criptográfica de propósito general. La biblioteca criptográfica de OpenSSL provee una implementación de protocolos de seguridad como SSL (Secure Sockets Layer) y TLS (Transport Layer Security) escrita en su mayoría en lenguaje C[4]. Además, la biblioteca criptográfica puede usarse libremente para propósitos comerciales y no comerciales bajo una licencia estilo Apache[2].

2.2. Protocolos SSL/TLS y DTLS

El protocolo TLS (Transport Layer Security) y su predecesor SSL (Secure Sockets Layer) son protocolos criptográficos que permiten la transferencia segura de datos a través de una red[2]. Las principales aplicaciones de estos protocolos son en navegadores web, correo electrónico y mensajería instantánea. Además, una gran cantidad de sitios web utilizan SSL/TLS para proteger todas las comunicaciones entre sus servidores y los navegadores web[5].

El protocolo DTLS (Datagram Transport Layer Security) es un protocolo criptográfico que proporciona privacidad en las comunicaciones para protocolos de datagramas (paquete de información). El protocolo permite que las aplicaciones de tipo cliente/servidor se comuniquen de manera que se previene el espionaje y la manipulación o alteración de mensajes. El protocolo DTLS está basado en el protocolo TLS y proporciona garantías de seguridad equivalentes[6].

2.3. La extensión Heartbeat para TLS/DTLS

La extensión Heartbeat para los protocolos TLS y DTLS se creó con la finalidad de checar que la conexión entre dos dispositivos aún está “viva”, esto es, que ambos dispositivos aun sean capaces de comunicarse entre sí. Tal como está indicado en el RFC[7], la extensión Heartbeat mantiene la conexión

entre dos pares, requiriendo que cada uno de ellos intercambie un mensaje especial al que se le llamó "heartbeat".

La razón por la que se decidió crear esta extensión no fue debido a que los protocolos TLS y DTLS no cuenten con alguna manera de determinar si la conexión aún está activa, sino por el costo de dichas verificaciones. Por un lado, el protocolo DTLS está diseñado para proteger el tráfico de información sobre protocolos de transporte no confiables (aquellos que no notifican al usuario si una transferencia falla) y el único mecanismo que posee para determinar si la conexión aún está activa consiste en una renegociación costosa. Por otro lado, el protocolo TLS, que está basado en protocolos de transporte confiables (aquellos que notifican al usuario si una transferencia falla), no cuenta con una forma de averiguar si la conexión aún está activa sin necesidad de estar transmitiendo datos continuamente[7].

3. Heartbleed

3.1. Funcionamiento interno de la extensión Heartbeat

El protocolo de extensión Heartbeat cuenta con dos tipos de mensajes: *heartbeat_request* (petición Heartbeat) y *heartbeat_response* (respuesta Heartbeat). En su implementación se definen mediante una enumeración como se muestra a continuación[7].

```
enum {
      heartbeat_request(1),
      heartbeat_response(2),
      (255)
} HeartbeatMessageType;
```

A su vez, los mensajes están implementados a partir de una estructura en lenguaje C[7] tal como se muestra a continuación.

```
struct {
          HeartbeatMessageType type;
          uint16 payload_length;
          opaque payload[HeartbeatMessage.payload_length];
          opaque padding[padding_length];
} HeartbeatMessage;
```

El propósito que cumplen los campos de esta estructura es el siguiente:

- ***type***: El tipo del mensaje, puede ser *heartbeat_request*, o bien, *heartbeat_response*
- ***payload_length***: La longitud del *payload*.
- ***payload***: Contenido arbitrario que será enviado en el mensaje.
- ***padding***: Contenido aleatorio que debe ser ignorado por el receptor. Debido a la implementación, su tamaño *padding_length* debe ser por lo menos de 16 bytes.

La forma en que estos mensajes se envían depende del tipo de protocolo de transporte que se esté utilizando[2]:

- Al usar un protocolo de transporte confiable (TLS):

  Desde un lado de la conexión entre pares (peer-to-peer) se envía un mensaje del tipo *heartbeat_request* hacia el otro lado. el receptor de la conexión debe enviar inmediatamente una respuesta *heartbeat_response*. Esta serie de eventos constituye un "Heartbeat" exitoso y la conexión se mantiene activa. No obstante, si no se recibe ninguna respuesta después de un tiempo de espera específico, la conexión TLS se termina.

- Al usar un protocolo de transporte no confiable (DTLS):

  De la misma manera que usando un protocolo confiable, desde un lado de la conexión se envía un mensaje del tipo *heartbeat_request* y el otro lado de la conexión debe enviar inmediatamente un mensaje *heartbeat_response*. La diferencia está en que si no se recibe una respuesta después de un tiempo de espera específico, el mensaje *heartbeat_request* vuelve a retransmitirse; y si no se recibe respuesta después de un número específico de retransmisiones, la conexión DTLS se termina.

Cuando el receptor recibe un mensaje del tipo *heartbeat_request*, éste mismo debe responder una copia del mensaje recibido como un *heartbeat_response*. El emisor del mensaje original verifica que el payload del mensaje de respuesta sea el mismo que originalmente envió. Si estos mensajes coinciden, la conexión se mantiene activa; si no coinciden, el mensaje *heartbeat_request* se retransmite un número específico de veces.

3.2. ¿Qué salió mal con la extensión Heartbeat?

Como tal, la mayor parte del código que se utilizó para implementar la extensión Heartbear es correcto. Sin embargo, la función encargada de crear los *heartbeat_response* contiene algunas líneas que

ocasionaron el error conocido como Heartbleed[2].

```
int tls1_process_heartbeat(SSL *s){
                                    .
                                    .
                                    .
        /* Allocate memory for the response, size is 1 byte
         * message type, plus 2 bytes payload length, plus
         * payload, plus padding
         */
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        bp = buffer;
        /* Enter response type, length and copy payload */
        *bp++ = TLS1_HB_RESPONSE;
        s2n(payload, bp);
        memcpy(bp, pl, payload);
        bp += payload;
        /* Random padding */
        RAND_pseudo_bytes(bp, padding);
        r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer,
                3 + payload + padding);
                                    .
                                    .
                                    .
        return 0;
}
```

El error del código mostrado se encuentra en el uso de la función *memcpy()* debido a que el código no realiza una verificación de que la longitud del payload recibido en el *heartbeat_request* corresponda con su longitud real. Esto ocasiona que si a la longitud del payload introducida se le asigna un valor más grande que la longitud real, la función *memcpy()* copiará el contenido del payload y cualquier otro contenido adyacente en memoria al final del payload.

La función *memcpy()* permite copiar a lo mucho 65535 bytes. Esto significa que un atacante puede copiar 65535 bytes de un dispositivo vulnerable y reenviarlos directo al dispositivo donde realizó la petición[2]. Si bien el atacante no puede elegir qué espacio en memoria copiará en un tiempo determinado, al explotar esta vulnerabilidad repetidas veces es posible que se filtre información confidencial como nombres de usuario, contraseñas, e incluso las llaves privadas de la encriptación[1]. La siguiente imagen muestra un ejemplo de dicha filtración de información.

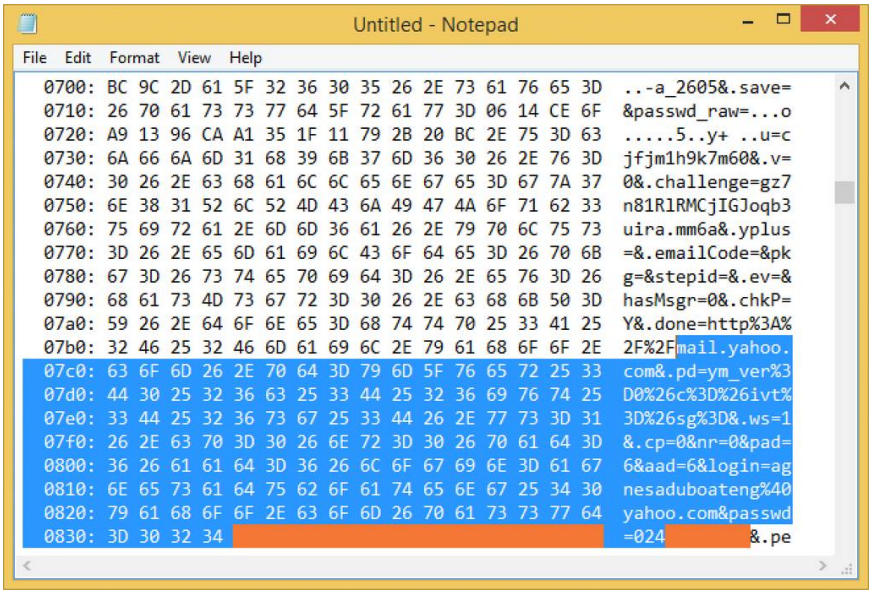

Figura 1: Ejemplo de información filtrada a través de Heartbleed.
Imagen tomada de: `http://www.redeszone.net/2014/04/09/como-comprobar-si-tu-web-es-vulnerable-heartbleed-el-fallo-de-openssl/`

3.3. Impacto real del Heartbleed

La Agencia de Ingresos de Canadá reportó el robo de números de Seguro Social que pertenecen a 900 contribuyentes. La agencia declaró que fueron accedidos a través de un exploit del fallo durante un período de 6 horas el 8 de abril de 2014[8].

En otro incidente, el sitio de crianza de los hijos del Reino Unido Mumsnet tuvo varias cuentas de usuario secuestradas, y se hicieron pasar por su director general[8].

Investigadores anti-malware explotaron Heartbleed para acceder a foros secretos utilizadas por ciber-delincuentes[8].

3.3.1. Dispositivos y sistemas operativos afectados

Hearbleed no solo afectó a la web, sino también a un gran número de sistemas embebidos y diversos sistemas operativos. A continuación se presenta una lista de los afectados[2]:

- Smartphones con Android 4.1.1 (Jelly Bean).
- Routers Cisco.
- Routers Juniper.

- Debian Wheezy (stable), OpenSSL 1.0.1e-2+deb7u4.
- Ubuntu 12.04 LTS, OpenSSL 1.0.1-4ubuntu5.11
- CentOS 6.5, OpenSSL 1.0.1e-15
- NetBSD 5.0.2, OpenSSL 1.0.1e
- FreeBSD 10, OpenSSL 1.0.1e
- Fedora 18, OpenSSL 1.0.1e-4
- OpenSUSE 12.2, OpenSSL 1.0.1c

Referencias

[1] Liang Zhang, "Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed." https://securepki.org/papers/Heartbleed-IMC.pdf, 2014. Recuperado el 24 de octubre de 2016.

[2] Bipin Chandra, "A technical view of theOpenSSL 'Heartbleed' vulnerability." https://ibm.biz/BdRCX7, 2014. Recuperado el 22 de octubre de 2016.

[3] Codenomicon, "The Heartbleed Bug." http://heartbleed.com/, 2014. Recuperado el 22 de octubre de 2016.

[4] T. Eric Young, "OpenSSL. Cryptography and SSL/TLS Toolkit." https://github.com/openssl/openssl, 2016. Recuperado el 24 de octubre de 2016.

[5] Wikipedia, "Transport Layer Security." https://en.wikipedia.org/wiki/Transport_Layer_Security, 2016. Recuperado el 24 de octubre de 2016.

[6] E. Rescorla, "Datagram Transport Layer Security Version 1.2." https://tools.ietf.org/html/rfc6347, 2012. Recuperado el 24 de octubre de 2016.

[7] R. Seggelmann, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension." https://tools.ietf.org/html/rfc6520, 2012. Recuperado el 25 de octubre de 2016.

[8] Wikipedia, "Hearbleed." https://en.wikipedia.org/wiki/Heartbleed, 2016. Recuperado el 24 de octubre de 2016.